PATENT ABSTRACTS OF JAPAN

(11)Publication number : **09-152961**
(43)Date of publication of application : **10.06.1997**

(51)Int.Cl. G06F 9/06
G06F 9/445

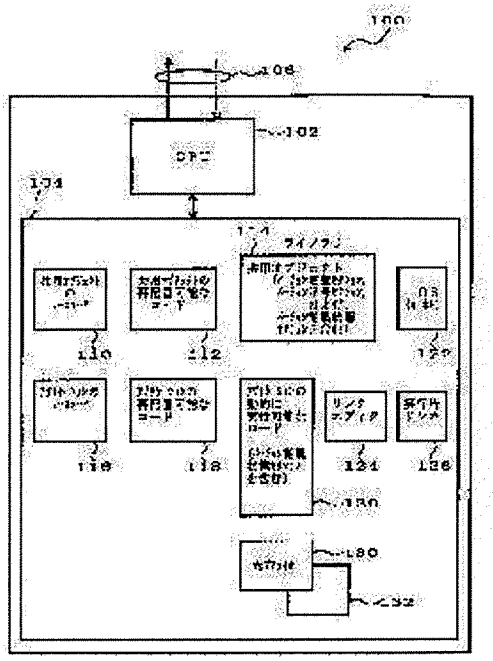
(21)Application number : **08-195380**
(22)Date of filing : **05.07.1996**

(71)Applicant : **SUN MICROSYST INC**
(72)Inventor : **EVANS RODRICK I**
GINGELL ROBERT A

(30)Priority
Priority number : **95 499062** Priority date : **06.07.1995** Priority country : **US**

(54) METHOD FOR IMPARTING VERSION SETTING INFORMATION TO SOFTWARE PROGRAM AND DEVICE THEREFOR

(57)Abstract:
PROBLEM TO BE SOLVED: To enable an easy and efficient version setting by confirming whether the object for which a version setting is performed is possible to be utilized as an execution time linker.
SOLUTION: At the time of an execution, an execution time linker 126 executes an object 120 which is possible to be dynamically executed by preparing a processing file 122 and executing the file when all the versions of a shared object 114 that the object 120 requires. Thus, a link editor 124 decides which version of the shared object is required for an application program. The execution time linker 26 maps the objects of the memory and couples the objects. Thus, the execution time linker 126 couples the objects just as the linker 26 is instructed by the link editor 124. Further, when a right version does not exist, the execution time linker 126 generates an error.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号

特開平9－152961

(43)公開日　平成９年(1997)６月10日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	9/06	４１０		Ｇ０６Ｆ　9/06	４１０Ｐ	
	9/445				４２０Ｃ	

審査請求　未請求　請求項の数３　ＦＤ　（全 22 頁）

(21)出願番号　特願平8－195380

(22)出願日　平成８年(1996)７月５日

(31)優先権主張番号　０８／４９９，０６２

(32)優先日　1995年７月６日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　595034134
サン・マイクロシステムズ・インコーポレイテッド
Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ，　Ｉｎｃ．
アメリカ合衆国カリフォルニア州94043－1100・マウンテンビュー・ガルシアアベニュー　2550

(72)発明者　ロドリック　アイ　エバンズ
アメリカ合衆国　94040　カリフォルニア，マウンテンビュー，ハンスアベニュー　784

(74)代理人　弁理士　飯塚　義仁

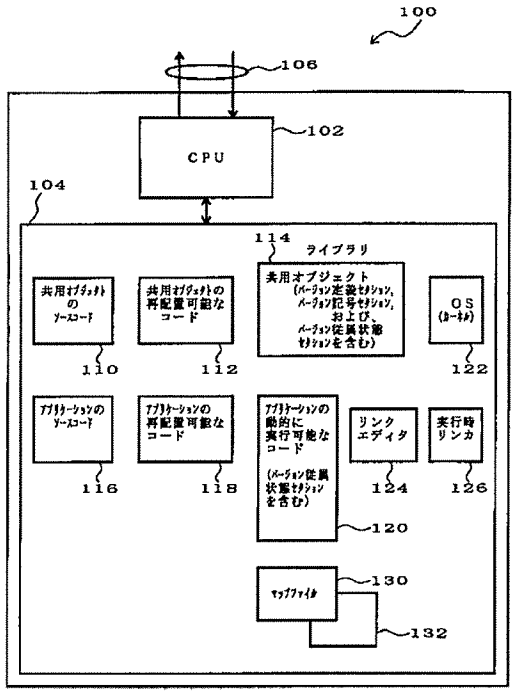
最終頁に続く

(54)【発明の名称】　ソフトウエアプログラムにバージョン設定情報を付す方法および装置

(57)【要約】

【課題】　ソフトウエアプログラムについて、簡単且つ効率的なバージョン設定を実現できるようにする。

【解決手段】　構築時にオブジェクトがコンパイルされ、リンクされる場合、リンクエディタは、前記オブジェクト内に、このオブジェクトの様々なバージョンに定義された大域記号を示すバージョン定義セクションおよびバージョン記号セクションを作成する。前記オブジェクトは、共用オブジェクト、再配置可能なオブジェクトまたは動的に実行可能なオブジェクトであってよい。構築時にアプリケーションソフトウエアプログラムがバージョン情報を有するオブジェクトとリンクされる場合、前記エディタは、その結果としての動的に実行可能なオブジェクトに、このオブジェクトのどのバージョンが前記プログラムの実行に必要であるかを示すバージョン従属状態セクションを作成する。実行時リンカは、前記プログラムの実行前に、前記オブジェクトの必要なバージョンのすべてが存在するかを判定する。



【特許請求の範囲】
【請求項１】　ソフトウエアプログラムにバージョン設定情報を付す方法であって、
ソフトウエアプログラムのためのオブジェクトコードを用意するステップと、
前記ソフトウエアプログラムのバージョンに関するバージョン名を示すマップファイルを用意するステップと、
前記ソフトウエアプログラムのバージョン名を示す情報が、前記マップファイルに従って前記オブジェクトコードに付加されるよう、前記オブジェクトコードをリンクし、これにより、バージョン設定されたオブジェクトを生成するステップとを具備し、これらのステップがデータ処理システムによって実行される方法。
【請求項２】　ソフトウエアプログラムにバージョン設定情報を付す方法であって、
第１のソフトウエアプログラムのための第１のオブジェクトコードを用意するステップと、
前記第１のソフトウエアプログラムのバージョンに関するバージョン名を示すマップファイルを用意するステップと、
前記第１のソフトウエアプログラムのバージョン名を示す情報が、前記マップファイルに従って前記第１のオブジェクトコードに付加されるよう、前記第１のオブジェクトコードをリンクし、これにより、バージョン設定されたオブジェクトを生成するステップと、
第２のソフトウエアプログラムのための第２のオブジェクトコードを用意するステップと、
前記第２のオブジェクトコードを前記バージョン設定されたオブジェクトにリンクするステップであって、このステップが、前記第２のソフトウエアプログラムに必要とされる前記第１のソフトウエアプログラムのバージョンを判定するステップと、前記第２のソフトウエアプログラムに必要とされる前記バージョンを示す情報を前記第２のオブジェクトコードに付加し、これにより、動的に実行可能なプログラムを生成するステップとをさらに含むものと、を具備し、これらのステップがデータ処理システムによって実行される方法。
【請求項３】　ソフトウエアプログラムにバージョン設定情報を付す装置であって、
第１のソフトウエアプログラムのための第１のオブジェクトコードを格納する記憶媒体と、
前記第１のソフトウエアプログラムのバージョンに関するバージョン名を指定するマップファイルを格納する記憶媒体と、
前記マップファイルに従って、前記第１のオブジェクトコードに対して、前記第１のソフトウエアプログラムの前記バージョンのバージョン名を定義する付加情報を付し、これにより、バージョン設定されたオブジェクトを生成するリンカと、
第２のソフトウエアプログラムのための第２のオブジェクトコードを格納する記憶媒体と、
前記第２のソフトウエアプログラムに必要とされるバージョンを示す付加情報を前記第２のオブジェクトコードに付すことによって、前記第２のオブジェクトコードを前記バージョン設定されたオブジェクトにリンクし、これにより、動的に実行可能なプログラムを生成するリンカとを具備した装置。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】この発明は、ソフトウエアプログラムをリンクする方法および装置に関し、特に、ソフトウエアプログラムの連続したバージョンの変更を管理する動的リンクシステムを提供する方法および装置に関する。
【０００２】
【従来の技術】ソフトウエア開発というものは絶えず現在進行中のプロセスといえるものである。ソフトウエアの最初のバージョンは、該ソフトウエアが書かれたときのタスクには十分であろうが、時が経過し、新たな特徴が追加されるのに伴って、アップグレードを必要とする。ソフトウエアアプリケーションが共用オブジェクト（“ライブラリ”とも言う）に結合（バインド）されるような場合、このソフトウエア開発プロセスは、特に問題を伴う。共用オブジェクトが更新または変更される場合、しばしば、該共用オブジェクトに対するインターフェイスも更新または変更される。さらに、共用オブジェクトに対するインターフェイスが変更されない場合でも、しばしば、前記共用オブジェクトによって実行される機能の一部に変化が生じる。
【０００３】
【発明が解決しようとする課題】従来のシステムのあるものは、実行時（ランタイム）に、アプリケーションソフトウエアプログラムを共用オブジェクトに動的にリンクする。このようなシステムにおいては、共用オブジェクトの新たなバージョンがリリースされる度に、該共用オブジェクトにアクセスするアプリケーションソフトウエアプログラムを慎重にチェックする必要がある。また、（前記共用オブジェクトに対するインターフェイスが同じであっても）前記共用オブジェクトの動作が変っていないか否かを判定するためにも、前記アプリケーションソフトウエアプログラムをチェックする必要がある。従来、このようなチェックは人手によって行われていた。インターフェイスの不整合によって生じるエラーは、しばしば、インターフェイスの不整合が見つかり、または、共用オブジェクトがそれまでと同じ動作を行わないことにより、アプリケーションソフトウエアプログラムの実行中にのみ発見される。このような場合必要なのは、特定のアプリケーションソフトウエアプログラムが共用オブジェクトのどのバージョンにリンクしようとするのかを判定する手段、および、前記共用オブジェク

トの必要なバージョンが実行時における動的リンク処理中に存在しているか否かをチェックする手段である。

【０００４】従来、前記共用オブジェクト自体のファイル名は、新たなバージョンごとに更新されていた。このため、リンク処理中には前記共用オブジェクトの最も新しいバージョンのみが存在し、この最も新しいバージョンは該オブジェクトの旧いバージョンとは完全に異なるファイル名を有することになる。この点につき、オブジェクトの新たなバージョンが作成される毎に該オブジェクトの名前を変えないようにすることが望ましい。従来のシステムは、しばしば、アプリケーションプログラムおよび共用オブジェクト（ライブラリ）を“１かたまりのシステム”としてリリースすることによって、バージョンチェックを回避しようとしている。すなわち、共用オブジェクトのバージョンの変更がなされたか否かに関わらず、新たなアプリケーションは、それが必要とするオブジェクトのすべてと共に出荷される。この点につき、前記システムにおける、バージョン変更に対処するために必要な部分のみをアップグレードできるようにすることが望ましい。

【０００５】この発明は上述の点に鑑みてなされたもので、オブジェクトの新たなバージョンが作成される毎に該オブジェクトの名前を変えないようにすることができ、且つ、バージョン変更に対処するために必要な部分のみをアップグレードできるようにすることにより、簡単且つ効率的なバージョン設定を可能にする、ソフトウエアプログラムにバージョン設定情報を付す方法および装置を提供することを目的とする。

【０００６】

【課題を解決するための手段】この発明は、ソフトウエアプログラムを動的にリンクする方法および装置を提供するものであり、また、アプリケーションソフトウエアプログラムによってアクセスされるオブジェクトの連続したバージョンにおける変更を把握するバージョンシステムを提供する。この発明は、バージョン設定されたオブジェクトにアクセスするためにアプリケーションソフトウエアプログラムによって使用されるインターフェースをチェックし、前記アプリケーションソフトウエアプログラムによる前記バージョン設定されたオブジェクトの無効バージョンにアクセスする試みを検出する。こうして、この発明は、アプリケーションソフトウエアプログラムとオブジェクトとの間の後方互換性を維持しながら、オブジェクトの制御された進化を可能にするものである。

【０００７】この発明は、記号インターフェースおよび実行上の変更が１つのオブジェクト内においてラベル表示できるようにする、バージョン設定システムを提供する。構築時において、リンクエディタは、バージョン設定されたオブジェクトに対して、該オブジェクトのすべての利用可能なバージョンを定義するデータ（バージョン定義セクションおよびバージョン記号セクション）を付加する。また、構築時において、リンクエディタは、ソフトウエアアプリケーションに対して、該アプリケーションのバージョン要件を定義するデータ（バージョン従属状態セクション）を付加する。実行時において、実行時リンカは、前記ソフトウエアアプリケーションの要件が前記オブジェクト自体に格納されたバージョン定義に一致するか否か、すなわち、前記ソフトウエアアプリケーションに必要とされる前記バージョン設定されたオブジェクトが前記実行時リンカに利用可能であるか否かを確認する。

【０００８】バージョンとは、オブジェクトに記録された名前またはラベルである。バージョンは１つまたは２つ以上の大域記号（グローバルシンボル）に関連づけられてよく、この場合、バージョンは記号インターフェースを定義する。そうでない場合、バージョンは、単に、実行上の変更、すなわち、オブジェクトの機能の変更が存在するものの、新たな大域記号の定義はなされない変更を示す標識であってよい。後者の場合、そのバージョンは“弱い（ウィーク）”バージョンと呼ばれる。ここに説明するこの発明の実施の形態において、各バージョンに関連した大域記号およびバージョン名は、人によって発生される“マップファイル”に定義される。バージョン設定されたオブジェクトを作成するために、前記マップファイルは、構築時に、１つまたは２つ以上の再配置可能な（コンパイルされた）オブジェクトと共に前記リンクエディタに入力される。構築時に、アプリケーションが（バージョン設定情報を有する）バージョン設定されたオブジェクトとリンク編集されるとき、デフォルト値により、前記アプリケーションによって参照される大域記号を含むバージョンに対する従属状態が前記アプリケーションに設定される。さらに、弱い定義に対する“弱い”従属状態が設定される。

【０００９】この発明は、１つのオブジェクト内におけるバージョン定義の継承を可能にする。バージョンはバージョンを受け継ぎ、これにより、相互に関連したインターフェース定義を作成するために、複数組の記号を組合わせることができる。例えば、１つの新たなバージョンは、旧いバージョンのすべての大域記号を受け継ぐことができる。この発明は、リンクエディット時におけるオブジェクトのバージョンの可視性を制御でき、実際上、前記アプリケーションプログラムに利用可能なインターフェースを制御できる。また、この発明は、前記アプリケーションプログラムがオブジェクトの弱いバージョンを要求することを強制することもできる。

【００１０】上記目的を達成するため、この発明は、ソフトウエアプログラムにバージョン設定情報を付す方法であって、ソフトウエアプログラムのためのオブジェクトコードを用意するステップと、前記ソフトウエアプログラムのバージョンに関するバージョン名を示すマップ

ファイルを用意するステップと、前記ソフトウエアプログラムのバージョン名を示す情報が、前記マップファイルに従って前記オブジェクトコードに付加されるよう、前記オブジェクトコードをリンクし、これにより、バージョン設定されたオブジェクトを生成するステップとを具備し、これらのステップがデータ処理システムによって実行されるものである。これにより、この発明によれば、バージョン設定されたオブジェクトを自動的に生成することができる。

【００１１】さらに、この発明は、ソフトウエアプログラムにバージョン設定情報を付す方法であって、第１のソフトウエアプログラムのための第１のオブジェクトコードを用意するステップと、前記第１のソフトウエアプログラムのバージョンに関するバージョン名を示すマップファイルを用意するステップと、前記第１のソフトウエアプログラムのバージョン名を示す情報が、前記マップファイルに従って前記第１のオブジェクトコードに付加されるよう、前記第１のオブジェクトコードをリンクし、これにより、バージョン設定されたオブジェクトを生成するステップと、第２のソフトウエアプログラムのための第２のオブジェクトコードを用意するステップと、前記第２のオブジェクトコードを前記バージョン設定されたオブジェクトにリンクするステップであって、このステップが、前記第２のソフトウエアプログラムに必要とされる前記第１のソフトウエアプログラムのバージョンを判定するステップと、前記第２のソフトウエアプログラムに必要とされる前記バージョンを示す情報を前記第２のオブジェクトコードに付加し、これにより、動的に実行可能なプログラムを生成するステップとをさらに含むものと、を具備し、これらのステップがデータ処理システムによって実行されるものである。これにより、この発明によれば、バージョン設定されたオブジェクトを自動的に生成することができると共に、第２のソフトウエアプログラムすなわちアプリケーションソフトウエアプログラムを、このバージョン設定されたオブジェクトに動的にリンクして実行可能とすることができる。

【００１２】別の観点に従えば、この発明は、ソフトウエアプログラムにバージョン設定情報を付す装置であって、第１のソフトウエアプログラムのための第１のオブジェクトコードを格納する記憶媒体と、前記第１のソフトウエアプログラムのバージョンに関するバージョン名を指定するマップファイルを格納する記憶媒体と、前記マップファイルに従って、前記第１のオブジェクトコードに対して、前記第１のソフトウエアプログラムの前記バージョンのバージョン名を定義する付加情報を付し、これにより、バージョン設定されたオブジェクトを生成するリンカと、第２のソフトウエアプログラムのための第２のオブジェクトコードを格納する記憶媒体と、前記第２のソフトウエアプログラムに必要とされるバージョンを示す付加情報を前記第２のオブジェクトコードに付すことによって、前記第２のオブジェクトコードを前記バージョン設定されたオブジェクトにリンクし、これにより、動的に実行可能なプログラムを生成するリンカとを具備したものである。

【００１３】別の観点に従えば、この発明は、動的に実行可能なオブジェクトに必要とされるオブジェクトのバージョンが該実行可能オブジェクトの実行中に存在することを判定させるための、コンピュータによって読み取り可能なコードを格納したコンピュータによって使用可能な媒体を備えたコンピュータプログラム製品であって、コンピュータに、第１のソフトウエアプログラムのための第１のオブジェクトコードを用意させる第１のコンピュータ読み取り可能プログラムコード装置と、コンピュータに、前記第１のソフトウエアプログラムのバージョンに関連したバージョン名を指定するマップファイルを用意させる第２のコンピュータ読み取り可能プログラムコード装置と、前記第１のソフトウエアプログラムのバージョン名を示す情報が、前記マップファイルに従って、前記第１のオブジェクトコードに付加されるよう、コンピュータに、前記第１のオブジェクトコードをリンクさせ、これにより、バージョン設定されたオブジェクトを生成させる第３のコンピュータ読み取り可能プログラムコード装置とを具備したものである。

【００１４】

【発明の実施の形態】以下、添付図面を参照してこの発明の一実施の形態を詳細に説明する。

１．概論

図１は、この発明に係るコンピュータシステム１００のブロック図である。該コンピュータシステム１００は、ＣＰＵ１０２と、メモリ１０４と、入出力ライン１０６とを備えている。当業者に理解されるように、前記コンピュータシステム１００は、ディスクドライブ、キーボード、ディスプレイ装置、ネットワーク接続部、付加的なメモリ、付加的なＣＰＵ等、ここでは明確さのために図示していないその他の多数の要素を含んでいてもよい。前記メモリ１０４は、ライブラリ（共用オブジェクトとも言う）１１４と、該共用オブジェクト１１４のソースコード１１０および再配置可能な（コンパイルされた）コード１１２とを格納している。さらに、前記メモリ１０４は、アプリケーションソフトウエアプログラムのソースコード１１６と、アプリケーション１１６の再配置可能な（コンパイルされた）コード１１８と、アプリケーション１１６の動的に実行可能な（リンクされた）コード１２０を格納している。また、前記メモリ１０４は、ＯＳ（カーネル）ソフトウエア１２２と、マップファイル１３０，１３２と、リンクエディタ１２４と、実行時（ランタイム）リンカ１２６とを格納している。前記共用オブジェクト１１４および実行可能オブジェクト１２０の各々は、前記リンクエディタ１２４によ

って作成されるものである。前記共用オブジェクト１１４および実行可能コード１２０の各々は、Prentice Hall, Inc.によって発行された“System V Application Binary Interface”の第３版に定義されたExecutable Linking Format（実行可能リンクフォーマット、以下、“ＥＬＦフォーマット”とも言う）と同様なＥＬＦを有する。

【００１５】しかしながら、以下に説明するように、この発明において、前記オブジェクト１１４，１２０のＥＬＦフォーマットは、付加的なデータを含むよう拡張されている。共用オブジェクト１１４は、図９に示すフォーマットを有する。前記動的に実行可能なオブジェクト１２０は、図１４に示すフォーマットを有する。以下に説明するように、前記共用オブジェクト１１４は、該共用オブジェクト１１４の各バージョン毎のバージョン設定情報（バージョン定義セクション）と、各バージョン毎の公用記号リスト（バージョン記号セクション）とを含んでいる。さらに、前記共用オブジェクト１１４は、バージョンの従属状態（関係）に関する情報（バージョン従属状態セクション）を含んでいてよい。実行可能オブジェクト１２０は、バージョンの従属状態（関係）に関する情報（バージョン従属状態セクション）を含んでいる。明確さのために図示されていないが、当業者に理解されるように、前記メモリ１０４は、アプリケーションプログラム、オペレーティングシステム、データ等のその他の情報をも格納している。

【００１６】この発明の好ましい実施の形態は、Solarisオペレーティングシステムのバージョン２．５の下に実施される。SolarisはSun Microsystems, Inc. の登録商標である。また、Unixは、X/OPEN, Ltd. により排他的にライセンスされ、米国およびその他の国で登録された商標である。図２は、図１のリンクエディタ１２４の入出力を示す図であり、バージョン設定された共用オブジェクトの作成を示す。以下では共用オブジェクトのバージョン設定について説明するが、この発明は、動的に実行可能なオブジェクトおよび再配置可能なオブジェクトのバージョン設定を実行するためにも使用可能である。このように、これらの種類のオブジェクトは、バージョン定義セクションおよびバージョン記号セクションを含むことができる。前記リンカエディタ１２４は、マップファイル１３０および共用オブジェクト用の再配置可能なオブジェクトコード１１２の入力を受け取り、共用オブジェクト１１４の出力を発生する。前記マップファイル１３０は、前記共用オブジェクトの各バージョン毎に大域（グローバル）記号およびバージョン名を指定する。ここに説明する実施の形態において、好ましくは、前記マップファイル１３０は、図８のフォーマットを有し、人によって作成される。他の実施の形態において、前記マップファイル１３０は、コンパイルシステムによって作成されてもよい。

【００１７】図３は、図１のリンクエディタ１２４の他の入出力を示す図であり、図２の共用オブジェクトによるアプリケーションプログラムのリンクエディト（編集）を示すものである。前記リンクエディタ１２４は、共用オブジェクト１１４および再配置可能なオブジェクト１１８を入力し、これらのオブジェクト１１４，１１８を処理することによって動的に実行可能なオブジェクト１２０を生成する。図４に示すように、前記オブジェクト１１４のどのバージョンがこのリンク手続きに許容されるかを示すために、このステップにマップファイル１３２が使用されてもよい。好ましくは、該マップファイル１３２は、図１３のフォーマットを有し、人によって作成される。

【００１８】図５に示すように、実行時において、実行時リンカ１２６は、動的に実行可能なオブジェクト１２０が必要とする共用オブジェクト１１４のすべてのバージョンが存在するか否かを確認する。前記すべてのバージョンが存在する場合、前記実行時リンカ１２６は、処理ファイル１２２を作成して実行することによって、動的に実行可能なオブジェクト１２０を実行する。こうして、前記リンクエディタ１２４は、再配置可能なオブジェクトからバージョン設定された共用オブジェクトを構築し、前記共用オブジェクトのどのバージョンがアプリケーションプログラムに必要とされるのかを判定する。前記実行時リンカ１２６は、そのメモリのオブジェクトをマップし、結合する。こうして、前記実行時リンカ１２６は、単に、前記リンクエディタ１２４によって指示されるようにオブジェクトを結合する。さらに、前記実行時リンカ１２６は、動的に実行可能なオブジェクト１２０が必要とする共用オブジェクト１１４のバージョンが存在することを保証するための、確認チェックを行う。正しいバージョンが存在しない場合、前記実行時リンカ１２６はエラーを発生する。

【００１９】次に、共用オブジェクト１１４の様々なバージョン間で可能な変更について説明する。一般に、これらの変更は、互換性のある更新と互換性の無い更新との２つのグループに分類できる。前記互換性のある更新とは、付加的な更新、すなわち、共用オブジェクト１１４に対するインターフェースでそれまで利用可能であった大域記号がそのまま維持される更新である。互換性のある更新の一例は、大域記号の付加である。前のバージョンから記号は除去されていないので、前のバージョンとインターフェース接続されたアプリケーションソフトウエアは依然として正しく動作する。前記互換性の無い更新は、特定の既存のインターフェースを使用した既存のアプリケーションが故障したり不正動作したりするような状態に、前記既存のインターフェースを変更するものである。このような互換性の無い更新の例としては、記号の除去、機能に対する引数の付加、機能からの引数の除去、および、機能に対する引数のサイズまたは内容

の変更などがある。共用オブジェクト１１４に対する誤り訂正は、既存のインターフェースに対する互換性のある更新であったり、互換性の無い更新であったりする。例えば、互換性のある誤り訂正は、単に、それまで定義されていたインターフェースを維持しながら、共用オブジェクト１１４の内部機能を変更するものである。一方、互換性の無い誤り訂正は、共用オブジェクト１１４に対するインターフェースの変更を必要とする。

【００２０】２．構築時におけるバージョン設定情報の作成

上記パラグラフは、この発明に従うバージョン設定を実行するために、構築時および実行時に行われる処理の全体的な説明であった。以下のパラグラフは、前記リンクエディタ１２４が、どのようにして、共用オブジェクトおよびアプリケーションプログラムにバージョン設定情報を付加するのかを説明するものである。

【００２１】ａ．構築時にバージョン設定されたオブジェクトに関する

バージョン定義作成

以下に説明するように、好ましい実施の形態において、前記リンクエディタ１２４は、マップファイル１３０のバージョン指示情報に従って、共用オブジェクトのバージョン設定を制御する。前記マップファイル１３０は、人によって作成されるのが好ましいが、コンパイルシステムのようなソフトウエアによって作成されてもよい。図６および図７は、図２のリンクエディタ１２４によって実行されるステップを示すフローチャートである。これらのステップは、共用オブジェクト１１４のようなバージョン設定されたオブジェクトを作成するために実行されるステップの一部である。当業者によって理解されるように、図６および図７（および図１２）のステップは、メモリ１０４に格納されていて、且つ、例えばメモリ１０４に格納されたデータ構造を使用したリンクエディタ１２４の命令を実行する図１のＣＰＵ１０２によって行われる。

【００２２】ステップ３０２に示すように、図６のステップは、リンクエディタ１２４がコマンドライン上の-Ｇオプションによって起動された時に開始される。前記-Ｇオプションは、リンクエディタ１２４が（動的に実行可能なオブジェクトとは対照的な）共用オブジェクトを生成すべきことを示すものである。この発明の好ましい実施の形態において、前記リンクエディタ１２４は-Ｍオプションによっても起動される。該-Ｍオプションは、マップファイル１３０が“バージョン定義指示”ソースとして使用されるべきことを示すものである。図２の例は共用オブジェクトの作成を示しているが、この発明は、バージョン設定された再配置可能なオブジェクトおよび動的に実行可能なオブジェクトを作成するためにも使用可能である。

【００２３】以下の例は、使用されるUnixコマンド（cat, cc, pvs, ld）の詳細を含んでいない。該Unixコマンドは、Sun Microsystemsから入手可能なSolaris ReferenceManualに記載されている。この後のパラグラフでは、共用オブジェクトソースコード１１０およびマップファイル１３０の一例について説明する。表１は、Ｃプログラミング言語で書かれた４つのソースファイル（“foo.c”，“data.c”，“bar1.c”および“bar2.c”）のソースコード１１０を示している。これらのソースコードファイルは、１つの共用オブジェクトを構成している。また、表２のマップファイルは、前記共用オブジェクトの様々なバージョンの大域インターフェースを定義している。これらのファイルは、コンパイルされることにより、図１の再配置可能なオブジェクト１１２を構成する。その後、前記リンクエディタ１２４は、後で説明するように、マップファイル１３０に従って、バージョン設定された共用オブジェクト１１４を作成する。

【００２４】

【表１】

$ cat foo.c

```
extern const char *  _fool;
extern const char *  _foo2;

void fool()
{
        (void) printf(_fool);
}

void foo2()
{
        (void) printf(_foo2);
}
```

$ cat data.c

```
const char * _foo1 = "string used by function foo1()\n";
const char * _foo2 = "string used by function foo2()\n";
```

$ cat bar1.c

```
extern void foo1();

void bar1()

{
        foo1();
}
```

$ cat bar2.c

```
extern void foo2();

void bar2()
{
        foo2();
}
```

【００２５】

【表２】

```
$ cat mapfile
SUNW.1.1 {                          #Release X
        global:
                foo1;
        local:
                *;
};

SUNW.1.2 {                          #Release X+1
        global:
                foo2;
} SUNW.1.1;

SUNW.1.2.1 { } SUNW.1.2;            #Release X+2

SUNW.1.3a {                         #Release X+3
        global:
                bar1;
} SUNW.1.2;

SUNW.1.3b {                         #Release X+3
        global:
                bar2;
} SUNW.1.2;

SUNW.1.4 {
        global:
                bar3;
} SUNW.1.3a SUNW.1.3b
```

【００２６】図８は、Backus-Naur（バッカスナウル）フォーマットを使用したマップファイル１３０のフォーマットを示すものである。該バッカスナウルフォーマットにおいて、“［”および“］”はオプショナル要素を示す。例えば、図８において、“［version name］”は前記マップファイルフォーマットのオプショナル要素である。共用オブジェクト１１４は、実行時において動的に実行可能なオブジェクト１２０のような他のオブジェクトが結合することが可能な大域記号を提供する。これらの大域記号は、前記マップファイル１３０において指定されており、共用オブジェクト１１４のアプリケーション・バイナリ・インターフェース（ＡＢＩ）を記述する。共用オブジェクト１１４の生存期間中、オブジェクトのインターフェースは、大域記号の付加または除去によって変更可能である。さらに、共用オブジェクト１１４の進化は、前記インターフェースの大域記号に影響しない、該オブジェクト１１４に対する内部実行上の変更を伴うことがある。

【００２７】表２は、マップファイル１３０の一例を示すものである。該表２において、前記マップファイル１３０は、共用オブジェクト１１４のバージョンであるSUNW.1.1, SUNW.1.2, SUNW.1.2.1, SUNW.1.3a, SUNW.1.3bおよびSUNW.1.4に関するバージョン定義を含んでいる。これらのバージョン定義は、該共用オブジェクト１１４についてそれまで定義されていたすべてのバージョン（およびそれらの大域記号）を含む。この例において、SUNW.1.1はRelease Xに含まれる共用オブジェクト１１４のバージョンであり、SUNW.1.2は、SUNW.1.1の大域記号を受け継いだ、Release X+1に含まれる共用オブジェクト１１４のバージョンであり、SUNW.1.2.1は、SUNW.1.2の大域記号を受け継いだ、Release X+2に含まれる共用オブジェクト１１４のバージョンである。なお、SUNW.1.2.1は、新たな大域記号を含んでいないので、“弱い（weak：ウィーク）”バージョンである。このバージョンは、前記共用オブジェクトにおける“実行上の”変更を示す。その他の前記バージョンは、“インターフェース”の変更を示す。SUNW.1.2.1は、前記共用オブジェクトの機能変更を有するが、インターフェース変更を有さないバージョンを示す。

【００２８】さらに、SUNW.1.3aは、SUNW.1.2の大域記号を受け継いだ、Release X+3に含まれる共用オブジェクト１１４のバージョンであり、SUNW.1.3bも、SUNW.1.2の大域記号を受け継いだ、Release X+3に含まれる共用オブジェクト１１４のバージョンである。しかし、SUNW.1.3bバージョンは、SUNW.1.3aに定義された大域記号“bar1”の代りに、大域記号“bar2”を定義している。SUNW.1.4は、大域記号“bar3”を定義するものであり、前記SUNW.1.3aおよびSUNW.1.3bの大域記号を引き継いでいる。

【００２９】表２のテーブルの例において、記号“foo1”は、SUNW.1.1バージョンの公用インターフェースに定義された唯一の大域記号である。特殊な“自動縮小化”命令（“*”）によって、前記共用オブジェクトのすべてのその他の大域記号は、該共用オブジェクトに対するインターフェースの一部とならないよう、局部的な有効範囲に縮小される。こうして、前記SUNW.1.1バージョンの公用インターフェースは、前記大域記号“foo1”に関連した、そのバージョンの内部バージョン定義によって構成される。

【００３０】

【表３】

```
$ cc -o libfoo.so.1 -M mapfile -G foo.c bar1.c bar2.c data.c
$ ln -s libfoo.so.1 libfoo.so
$ pvs -dsv libfoo.so.1
        libfoo.so.1:
                _end;
                _GLOBAL_OFFSET_TABLE_;
                _DYNAMIC;
                _edata;
                _PROCEDURE_LINKAGE_TABLE_;
                _etext;
        SUNW.1.1:
                foo1;
                SUNW.1.1;
        SUNW.1.2:               {SUNW.1.1};
                foo2;
                SUNW.1.2;
        SUNW.1.2.1 [WEAK]:      {SUNW.1.2};
                SUNW.1.2.1;
        SUNW.1.3a:              {SUNW.1.2};
                bar1;
                SUNW.1.3a;
        SUNW.1.3b:              {SUNW.1.2};
                bar2;
                SUNW.1.3b;
        SUNW.1.4:               {SUNW.1.3a SUNW.1.3b};
                bar3;
                SUNW.1.4;
```

【００３１】この表３は、表１のソースコードファイルをコンパイルし、リンクするためのUnixコマンドを示している。オブジェクトコードファイル“foo.o”，“data.o”，“bar1.o”および“bar2.o”（図示せず）は、表２のマップファイル１３０を使用して動的にリンクされ、これにより、“libfoo.so.1”と称する共用オブジェクト１１４を生成する。（この例の場合、ccコンパイラは、自動的にld(1)リンクエディタ１２４を呼出す）。コマンドライン上の-Gオプションは、リンクエディタ１２４が動的に実行可能なオブジェクトではなく、共用オブジェクトを生成すべきことを示す。“ln”コマンドは、ld(1)の“-l”オプションに適した“コンパイル環境”名を作成する。“pvs”Unixコマンドは、前記リンクエディタ１２４によって作成される共用オブジェクトのバージョン設定情報、および、各バージョンに利用可能な大域記号をプリントアウトする。

【００３２】表３に示すように、前記オブジェクトについて“基準バージョン”の定義も作成される。この基準バージョンは、前記共用オブジェクト自体の名前（例えば“libfoo.so.1”）を使用して定義され、前記リンクエディタ１２４によって生成され確保された記号を前記オブジェクトに関連づける。例えば、表３の例において、基準バージョンの定義は、前記リンカによって作成され確保された大域記号（例えば、_etext, _edata, _end, _DYNAMIC, PROCEDURE_LINKAGE_TABLEおよびGLOBAL_OFFSET_TABLE）を含む共用オブジェクトlibfoo.so.1について作成される。

【００３３】図６に戻り、（ステップ３０４において）バージョン名がマップファイル１３０に現れた場合、前記リンクエディタ１２４は、共用オブジェクト１１４内に、特にバージョンに関する幾つかのセクションを作成する。これらの特別なセクションは、図９に示されており、バージョン定義セクション５０６、バージョン記号セクション５０８およびオプショナルのバージョン従属状態セクション５１０を含む。以下のパラグラフは、これらのセクションの作成および使用について説明するものである。

【００３４】図１０は、バージョン設定された共用オブジェクト１１４のバージョン定義セクション５０６のフォーマットを示すものである。また、図２０は、表１～表３に基づくバージョン定義セクションの一例を示すものである。該バージョン定義セクションは、セクションヘッダ６０２（図１６参照）と、構造セクション６０４と、フラグ６０６と、バージョン定義索引６１０と、カウント値６１２と、ハッシュ値６１４と、補助値６１６と、次バージョン定義ポインタ６１８と、該バージョン自体の名前６２０と、該定義されたバージョンが大域記号６２０，６２２を受け継いだバージョンの名前とを含んでいる。

【００３５】各共用オブジェクト１１４は、多数のバージョン定義を有する１つのバージョン定義セクション５０６を含んでいる。各フィールド６０４～６２２を、“バージョン定義”と言う。次のパラグラフは、バージョン定義の内容を説明するものである。前記セクションヘッダ６０２は、前記バージョン定義セクションに含まれるバージョンの数を示す。各バージョン定義セクションは、マップファイルに明示的に定義されていない大域記号を定義する基準バージョンに関する情報を含んでいる。このため、基準バージョンについてのフィールド６０４～６２２（バージョン定義）が、図６のステップ３０６で作成される。フィールド６０９（“基準”フラグ）は、この基準バージョン定義において設定される。

【００３６】ステップ３０８において、前記マップファイル１３０に定義された共用オブジェクト１１４の各バージョンごとに、バージョン定義（フィールド６０４～６２２）が、リンクエディタ１２４によって作成される。こうして、ステップ３０８において、表２のマップファイル１３０に関して、前記基準バージョン定義の他に６つのバージョン定義が作成される。図７のステップ３１０～３１６は、ステップ３０６，３０８で作成された各バージョン定義毎に実行される。ステップ３１０に示すように、前記マップファイル１３０が１つのバージョン（例えば、表１のSUNW.1.2.1参照）に関する大域記号を定義していない場合、そのバージョン定義に関する“ウィーク”フラグ６０８が、ステップ３１２において設定される（図２０参照）。

【００３７】フィールド６０４は、その構造自体のバージョン番号である。バージョン定義索引６１０は、共用オブジェクト１１４について定義された各バージョン毎に異なる（固有の）値を有するものであり、図１１との関係で後で説明することにする。カウント値６１２は、当該バージョンにおけるフィールド６２０および６２２の対のインスタンスの数を示すものである。ハッシュ値６１４は、このバージョンの名前に対するハッシュ値であり、通常のＥＬＦ（実行可能フォーマット）ハッシュ関数を使用する。また、補助値６１６は、このバージョンの第１のフィールド６２０に対する索引である。次バージョン定義６１８は、前記バージョン定義セクションにおける次のバージョン定義のフィールド６０４に対する索引である。

【００３８】ステップ３１３は、このバージョンの名前に対するエントリ６２０，６２２を作成し、このエントリを指し示すよう前記補助値６１６を設定する。こうして、前記第１のフィールド６２０は、前記バージョン自体のバージョン名を含んでいる。継承（受け継ぎ）情報は、定義されたバージョンが他のバージョンを受け継ぐ元となる１つまたは２つ以上のバージョンの名前で構成されている。図７のステップ３１４において、マップファイル１３０が図８および表２に示すように継承情報を有するか否かを調べる。ステップ３１６において、前記リンクエディタ１２４は、継承情報を保持する１つまたは２つ以上のエントリ６２０，６２２を作成する。該フィールド６２０，６２２は、特定のバージョンが他のバージョンを受け継ぐ元となるバージョン毎に存在する。前記フィールド６２０は受け継がれたバージョン定義の名前を含み、前記フィールド６２２は次のフィールド６２０（またはゼロ）を指し示す。

【００３９】図６のステップ３１８において、前記リンクエディタ１２４は、リンク編集中のオブジェクトのすべての記号について、バージョン記号セクションを作成する。図１１は、バージョン記号セクション５０８のフォーマットを示す図。前記バージョン記号セクション５０８におけるエントリは、そのバージョンに関する記号テーブル内の記号と１対１に対応する（図２１参照）。前記記号テーブルは、当業者によく知られており、前記System V Application Binary Interface Manualに記載されており、従って、ここでは説明しない。前記セクションヘッダ７０２のフォーマットは、図１７との関係において説明する。前記バージョン記号セクション５０８のエントリ７０４は、記号が定義されたバージョンの索引である。表２において、SUNW.1.3bバージョンが“６”の索引６１０を有する場合（図２０参照）、大域記号“bar2”に対応するバージョン記号セクションのエントリが“６”のエントリ値を有することになる（図２１）。図１１に示すように、局部的な有効範囲を有する記号のエントリは、“０”の値を有する。基準バージョン定義における記号のエントリは、“１”の値を有する。上述の如く、大域記号として明示的に定義された記号（例えば、基準セクションに関する記号、各バージョンの名前、“foo1”，“foo2”，“bar1”，“bar2”および“bar3”）のみが、バージョン記号セクション５０８においてゼロではないエントリ７０４を有する。図６のステップ３２０は、（必要な場合）共用オブジェクト１１４に関するバージョン従属状態セクションを作成する。例えば、共用オブジェクト１１４は、他のバージョン設定されたオブジェクトを参照することができる。このバージョン従属状態セクションの作成については、後で説明する。

【００４０】ｂ．構築時におけるバージョン従属状態情報の作成

表４は、アプリケーションプログラムソフトウエア１１６（“prog.c”）のソースバージョンの一例を示すものである。“prog.c”は、共用オブジェクト１１４libfoo.so.1の２つの大域記号、すなわち、“foo1”および“foo2”を参照する。これらの記号は、それぞれ、インターフェイスSUNW.1.1およびSUNW.1.2の一部として定義される。表４によると、構築時において、コンパイラccはldリンクエディタ１２４を起動する。構築時において、“prog.c”とバージョンSUNW.1.1，SUNW.1.2，SUNW.1.2.1間で、大域記号“foo1”および“foo2”を含む結合（バインディング）が行われる。前者の２つのバージョンは、記号結合を示す。後者のバージョンは、その弱い性質のため記録される。コンパイル／リンクコマンドによってバージョン制御指示は与えられないので、前記リンクエディタ１２４は、prog.cにおける大域記号を分解するときに存在する共用オブジェクト１１４のすべてのバージョンをチェックする。

【００４１】

【表４】

```
$ cat prog.c
extern void foo1();
extern void foo2();

main ()
{
        foo1();
        foo2();
}
$ cc -o prog prog.c -L. -R. -lfoo
$ pvs -r prog
        libfoo.so.1 (SUNW.1.2 SUNW.1.2.1);
```

【００４２】図１２は、再配置可能なオブジェクト１１８および共用オブジェクト１１４から動的に実行可能なＥＬＦファイル１２０を作成するために、構築時に前記リンクエディタ１２４によって実行されるステップを示すものである。前記動的に実行可能なＥＬＦファイル１２０のフォーマットは、図１４に示されている。図１４のフォーマットは、バージョン従属状態セクションを含むがバージョン定義セクションまたはバージョン記号セクションを含まないという点を除き、図９のフォーマットと同様である。図１５は、バージョン従属状態セクションのフォーマットを示す図である。

【００４３】図１２のステップ８０２において、前記リンクエディタ１２４は、リンク中のオブジェクト（例えばprog）における未分解の大域記号を、リンク中の他のオブジェクト（例えばlibfoo1.so.1）の大域記号テーブルと照会することによって、前記オブジェクトprogが前記他のオブジェクトに従属しているか否かを判定する。従属している場合、ステップ８０４において、前記リンクエディタ１２４は、必要とされるオブジェクトが複数のバージョンを有するか否か、すなわち、前記必要とされるオブジェクトがバージョン定義セクションを有するか否かを判定する。ステップ８０６において、前記必要とされるオブジェクトのいくつかのバージョンのみが前記リンクエディタ１２にとって可視性のものと判定された場合、制御はステップ８１０に進む。そうでない場合、制御はステップ８０８に進む。

【００４４】ステップ８０８において、前記リンクエディタ１２４は、共用オブジェクト１１４のすべての利用可能なバージョン定義セクションを調べ、どのバージョンが再配置可能なオブジェクト１１８に必要な大域記号を含むかを判定することによって、動的に実行可能なオブジェクト１２０にバージョン従属状態セクションを作成する。または、マップファイル１３２が、いくつかのバージョンのみを、前記リンクエディタ１２４にとって可視性のものとして識別してもよい。ステップ８１０では、前記リンクエディタ１２４は、共用オブジェクト１１４の可視性のバージョン定義セクションを調べ、どのバージョンが再配置可能なオブジェクト１１８に必要な大域記号を含むのかを判定する。

【００４５】図１４のバージョン定義セクションに記録される従属状態は、アプリケーションプログラムが共用オブジェクトに対するインターフェイスの大域記号を参照する毎に作成される。表４は、アプリケーションプログラムソフトウエア１１６（“prog.c”）のソースバージョンの一例を示すものである。該アプリケーションプログラム“prog.c”は、SUNW.1.1バージョンに定義された大域記号“foo1”およびSUNW.1.2バージョンに定義された大域記号“foo2”（表２参照）を参照する。こうして、アプリケーションプログラムprog.cと（SUNW.1.1バージョンから受け継いだ）SUNW.1.2バージョンとの間には、従属関係が存在する。前記リンクエディタ１２４は、prog.cの動的に実行可能なオブジェクト１２０に、prog.cがSUNW.1.2バージョンに従属していることを示すバージョン従属状態セクションを作成する（図２２参照）。前記再配置可能なオブジェクト１１８における未定義の大域記号毎に、前記リンクエディタ１２４は、前記再配置可能なオブジェクト１１８とリンクされているすべての共用オブジェクトのバージョン定義セクションおよびバージョン記号セクションを調べることによって、前記記号がどのバージョンに属するものかについての情報を得る。

【００４６】図１５は、バージョン定義セクション５１０のフォーマットを示すものである。該バージョン定義セクション５１０は、図１６に関連して説明するセクションヘッダ１１０２と、構造バージョン１１０４と、カウント値１１０６と、ファイル名１１０８と、補助値１１１０と、次バージョン従属状態セクション１１１２と、複数のフィールド１１１４～１１２２のインスタンスとを含んでいる。前記フィールド１１１４～１１２２は、ハッシュ値１１４と、“ウィーク”フラグ１１１６と、未使用フィールド１１１８と、名前フィールド１１２０と、次名前フィールド１１２２とを含んでいる。

【００４７】図１０に関連して上述したように、構造バージョン値１１０４はこの発明のバージョン設定に関係しない。カウント値１１０６は、フィールド１１１４～１１２２のインスタンスの数を示す。ファイル名１１０８は、共用オブジェクトの従属状態の名前である。補助値１１１０は、このバージョンに関する第１のフィールド１１１４に対する索引である。次バージョン従属状態セクション値１１１２は、次のバージョンフィールド１１０４に対する索引（またはゼロ）である。前記フィールド１１１４～１１２２の各インスタンスは、作成中のオブジェクトが必要とするバージョンを示す。バージョンが前記オブジェクトによって参照される大域記号を定義する場合、該バージョンが必要になる。常に、前記フィールド１１１４～１１２２の少なくとも１つのインスタンスが存在する。

【００４８】ハッシュ値１１４は、バージョン名１１２０から、通常のＥＬＦハッシュ関数を使用して発生される。“ウィーク”フラグ１１１６は、被従属（必要とされる）バージョンが弱いバージョンか否か、すなわち、

該バージョンが大域記号を含んでいないものか否かを示す。フィールド１１１８は、この発明のすべての実施の形態において形成されるものではなく、単に位置合せのために含まれる。ネームフィールド１１２０の第１のインスタンスは、前記バージョン自体の名前（例えば、“SUNW.1.2”）を含むものである。次名前フィールド１１２２は、次のハッシュ値１１１４を示すポインタである。

【００４９】この発明のこの実施の形態は、すべての継承（受け継いだ情報）がバージョン定義セクション５１０に記録されないようにする、“バージョン縮小化”を使用する。弱バージョンおよおよび必要とされるバージョンが、記録される。しかし、これらのバージョンが他のバージョンから大域記号を継承するとき、一般に、前記他のバージョンは記録されない。例えば、第１の弱いバージョンからの第２のバージョンを介した継承は、フィールド１１２０，１１２２に記録される前に、縮小化されるのが好ましい。同様に、第１の弱くないバージョンからの第２の弱くないバージョンを介した継承も、記録される前に縮小化されるのが好ましい。弱いバージョンと弱くないバージョンとの間の継承は、縮小化されない。こうして、SUNW.1.2はフィールド１１２０，１１２２に記録されるが、SUNW.1.1からの継承は縮小化されない。さらに、弱いバージョンであるSUNW.1.2.1に対する従属状態が記録される。

【００５０】図１６～図１８は、図１０、図１１および図１５のセクションの様々な様相を示すものである。図１６～図１８のフィールドは、従来のＥＬＦフォーマットの一部であるので、ここではこれらのすべては説明せず、この発明に関係するフィールドのみについて説明する。図１６は、セクションヘッダ６０７，７０２，１１０２に使用されるセクションヘッダのフォーマットを示す図である。“sh_name”フィールド１２０２はセクションの名前を含み、“sh_type”フィールド１２０４は前記セクションの種類を含んでいる。前記フィールドの値１２０２は、特にこの発明に関連のあるものである。図１７は、前記フィールド１２０４の様々な種類を表す値を示すものである。値１３０２は、特にこの発明に関連のあるものである。図１６のセクションヘッダフォーマットは、さらに、前記セクションにおけるバイト数を示す“sh_size”フィールド１３０６と、“sh_link”フィールド１３０８と、“sh_info”フィールド１３１０とを含んでいる。図１８は、前記“sh_type”，“sh_link”および“sh_info”の値の例を示している。値１４０２は、特にこの発明に関連のあるものである。

【００５１】上記パラグラフは、構築時における、バージョン定義セクション５０６およびバージョン記号セクション５０８を含むＥＬＦオブジェクト１１４の作成、および、バージョン従属状態セクション５１０を含むＥＬＦオブジェクト１２０の作成について説明した。なお、前記バージョン定義セクション５０６およびバージョン記号セクション５０８は大域記号の定義を含むオブジェクトのみに作成されるものであり、バージョン従属状態セクション５１０は、大域記号を含むオブジェクトに従属するオブジェクトのみに作成されるものである。こうして、大域インターフェースを有する共用オブジェクトは、セクション５０６，５０８および５１０を含むことになる。前記共用オブジェクトを参照する実行可能なアプリケーションプログラムは、バージョン従属状態セクション５１０のみを含むことになる。

【００５２】ｃ．構築時におけるアプリケーション構造の制限

上記パラグラフは、共用オブジェクト１１４のすべての利用可能なバージョンと再配置可能なオブジェクトとの間のバージョン結合の一例について説明した。結合は、アプリケーションプログラムと、図１３のオブジェクトの特定のバージョンのみとの間で発生するよう制限されることもできる。図１３は、図４のマップファイル１３２におけるファイル制御指示のフォーマットを示すものである。このフォーマットのファイル制御指示は、他のオブジェクトの特定のバージョンに従って動的に実行可能なオブジェクトを結合するために使用される。

【００５３】

【表５】

```
$ cat mapfile
libfoo.so - SUNW.1.1;

$ cc -o prog prog.c -M mapfile -L. -R. -lfoo
Undefined                       first referenced
 symbol                             in file
foo2                            prog.o (symbol belongs to \
unavailable version ./libfoo.so (SUNW.1.2))
ld: fatal: Symbol referencing errors.  No output written to prog
```

【００５４】図４および表５に示すように、前記リンクエディタが図１３に示されたファイル制御を含むマップファイル１３２を使用して実行される場合、前記オブジェクトの特定のバージョンについてのみ結合が行われる。表５において、マップファイルは、“libfoo.so-SUNW.1.1.”を含んでいる。表４に示したように、アプリケーションソフトウエア“prog.c”は、大域記号“foo1”および“foo2”を参照する。記号“foo2”は、SUNW.1.2バージョンに定義されている。（ccコンパイラおよびld（1）リンクエディタを使用して）prog.cが-Mオプションとリンクされる場合、該prog.cは、SUNW.1.1バージョンのみとリンクされる。このようにして、表５に示

すように、リンクエディタ１２４は、SUNW.1.2バージョンに定義された“foo2”を未定義の記号として認識する。

【００５５】ｄ．構築時における弱いバージョンの昇格

表６は、リンクエディタ１２４が弱いバージョン（SUNW.1.2.1）を強いバージョンに“昇格”するよう強制される例を示すものである。“-u”オプションにより、リンクエディタ１２４は、SUNW.1.2.1バージョンに対する“prog”の従属状態を、該“prog”のバージョン従属状態セクションに記録する。さらに、“ウィーク”フラグ１１１６が、SUNW.1.2.1バージョンに関して、“prog”のバージョン従属状態セクションにおいて、“偽”に設定される。この場合、SUNW.1.2.1バージョンはSUNW.1.2バージョンを受け継ぐ。故に、progの従属状態のすべては“強”であるので、バージョンの縮小化は、SUNW.1.2.1バージョンのみが弱くないバージョンとしてprogに記録される、ことを意味する。このような弱いバージョンの昇格は、“prog”が実行されるとき、それまで弱いものであったバージョンが存在することが実行時リンクエディタ１２４によって確認される、ことを保証する。

【００５６】

【表６】

```
$ cc -o prog prog.c -L. -R. -u SUNW.1.2.1  -lfoo

$pvs - r prog
      libfoo.so.1 (SUNW.1.2.1)
```

【００５７】３．実行時におけるバージョン設定情報の確認

図１９は、動的に実行可能なオブジェクト１２０をリンクして実行するときに、参照されるオブジェクトの必要なバージョンのすべてが存在することを保証するために、実行時リンカ１２６（図５参照）によって実行されるステップを示すフローチャート１５００である。図１９のステップは、好ましくは、ＣＰＵ１０２がメモリに格納された命令を実行することによって行われる。図１９のステップは、必要な従属状態のすべてが存在しているか否かを判定するために、実行前チェックとして行われる。

【００５８】ステップ１５０２において、前記リンカ１２６は、実行中の動的に実行可能なオブジェクト１２０がバージョン従属状態セクション（図１５参照）を含んでいるか否かを判定する。この判定結果がＮＯである場合、なんらチェックはなされず、通常の実行が続行される。この判定結果がＹＥＳである場合、ステップ１５０４において、前記リンカ１２６は、リンク中のオブジェクト１１４の少なくとも１つがバージョン定義セクションおよびバージョン記号セクションを含んでいるか否かを判定する。この判定結果がＮＯである場合、それ以上のチェックはなされず、通常の実行が続行される。この判定結果がＹＥＳである場合、制御はステップ１５０６に進む。

【００５９】ステップ１５０６において、前記実行時リンカ１２６は、現在の動的に実行可能なオブジェクト１２０におけるすべてのバージョン従属状態が処理されたか否かを判定する。すべてのバージョン従属状態が処理された場合、通常の実行が続行され、そうでない場合、制御はステップ１５０８に進む。ステップ１５０８では、前記実行時リンカ１２６は、（前記動的に実行可能なオブジェクト１２０のバージョン従属状態セクション）必要とされるバージョンと、前記共用オブジェクト１１４のバージョン定義セクションとが一致するか否かを判定する。例えば、図２０および図２２において、SUNW.1.2バージョンおよびSUNW.1.2.1バージョンは、共用オブジェクト１１４のバージョン定義セクションに定義されており、必要に応じて、動的に実行可能なオブジェクト１２０のバージョン従属状態セクションにおいて指定される。

【００６０】ステップ１５０８において一致が検出された場合、共用オブジェクト１１４の必要とされるバージョンが存在していることになり、ステップ１５０８において一致が検出されなかった場合、必要とされるバージョンが存在していないことになるので、ステップ１５１０において、その存在していないバージョンが弱いバージョンであるか、または、弱くないバージョンであるかを判定することが必要になる。前記実行時リンカ１２６は、動的に実行可能なオブジェクト１２０のバージョン従属状態セクションの“ウィーク”フラグ１１１６をチェックすることによって、前記バージョンが弱いものか否かを判定する。（なお、前記フラグ１１１６は、バージョンの弱から強への昇格を反映してもよい）。前記実行時リンカ１２６が、ステップ１５１０において、そのバージョンが弱いバージョンであると判定した場合、エラーは発生せず、制御はステップ１５０６に戻る。弱いバージョンではない判定した場合、必要とされる弱くないバージョンが存在していないので、致命的なエラーが発生する。従って、存在していない“弱い”バージョンへの従属はエラーを発生しないが、“弱くない”バージョンへの従属はエラーを発生することになる。

【００６１】図２０は、バージョン定義セクション１６００の一例を示し、図２１は、共用オブジェクト１１４（表１～表３のlibfoo.so.1）に関するバージョン記号セクション１７００の一例を示す。さらに、図２２は、アプリケーションプログラム１２０（表４の“prog”）の動的に実行可能な形態に関するバージョン従属状態セクション１８００の一例を示すものである。（この例において、共用オブジェクト１１４は、他のオブジェクトに従属しており、従って、バージョン従属状態セクションを有さない）。

【００６２】図２０は、基準バージョン定義１６０４、

および、それぞれのバージョンSUNW.1.1，SUNW.1.2，SUNW.1.2.1，SUNW.1.3a，SUNW.1.3bおよびSUNW.1.4のための６つのバージョン定義を含んでいる。前記基準バージョン定義１６０４の“基準”フラグは、“真（true）”に設定されている。また、SUNW.1.2.1バージョンに関するバージョン定義の“弱”フラグは、“真（true）”に設定されている。各バージョン定義は、固有のバージョン定義索引を有する。バージョンの縮小化は、バージョン定義テーブルにおいて適用される。アプリケーション“prog”は、（SUNW.1.1の）大域記号foo1および（SUNW.1.2の）大域記号foo2を参照する。SUNW.1.2はSUNW.1.1の大域記号を受け継ぐので、前記リンクエディタ１２４はバージョン縮小化を適用し、（SUNW.1.2のための）エントリがバージョン定義セクションにおいてなされる。弱いSUNW.1.2.1バージョンも記録される。

【００６３】図２１は、バージョン記号セクション１７００における記号のうちのいくつかを示す図である。例えば、大域変数“foo1”は、SUNW.1.1バージョンに定義されており、バージョン定義索引“２”を有する。こうして、foo1に関する記号テーブルのエントリに対応する前記バージョン記号セクションのエントリは、“２”を含む。上記実施の形態において、各バージョン（例えば、SUNW.1.1）の名前は、そのバージョン定義を生成するときに作成される大域記号でもある。

【００６４】図２２は、動的に実行可能なオブジェクト１２０におけるバージョン従属状態セクションの一例を示す図である。バージョン縮小化により、前記セクションは、SUNW.1.1ではなく、SUNW.1.2のためのエントリを含む。該セクションは、“弱い”SUNW.1.2.1バージョンのためのエントリをも含む。実行時において、リンカ１２６は、progがバージョン従属状態セクションを有すると判定し（ステップ１５０２）、libfoo.so.1がバージョン定義セクションを有すると判定し（ステップ１５０４）、必要とされるバージョン（SUNW.1.2）が存在すると判定する（ステップ１５０８）。故に、“prog”が実行されることになる。以上、いくつかの好ましい実施例について本発明につき説明したが、これら実施例に限らず、本発明の精神と範囲を逸脱しない限り、その他の種々の変形が可能であることが理解されるであろう。

【００６５】最後に、本願に係わる発明及びその実施態様のいくつかを要約して示すと下記の通りである。

（１）　ソフトウエアプログラムにバージョン設定情報を付す方法であって、第１のソフトウエアプログラムのための第１のオブジェクトコードを用意するステップと、前記第１のソフトウエアプログラムのバージョンに関するバージョン名を示すマップファイルを用意するステップと、前記第１のソフトウエアプログラムのバージョン名を示す情報が、前記マップファイルに従って前記第１のオブジェクトコードに付加されるよう、前記第１のオブジェクトコードをリンクし、これにより、バージョン設定されたオブジェクトを生成するステップとを具備し、これらのステップがデータ処理システムによって実行される方法。

（２）　第２のソフトウエアプログラムのための第２のオブジェクトコードを用意するステップと、前記第２のオブジェクトコードを前記バージョン設定されたオブジェクトにリンクするステップとをさらに具備し、前記リンクするステップが、前記第２のソフトウエアプログラムに必要とされる前記第１のソフトウエアプログラムのバージョンを判定するステップと、前記第２のソフトウエアプログラムに必要とされる前記バージョンを示す情報を前記第２のオブジェクトコードに付加し、これにより、動的に実行可能なプログラムを生成するステップとをさらに含むことを特徴とする前記１項に記載の方法。

（３）　前記動的に実行可能なプログラムを実行する前に、該実行可能なプログラムにおいて必要とされるバージョンが、前記バージョン設定されたオブジェクトにおいて示されるバージョンと一致するか否か、を判定するステップと、前記動的に実行可能なプログラムを実行するステップとをさらに具備し、前記動的に実行可能なプログラムが前記バージョン設定されたオブジェクトの前記必要とされるバージョンを呼出す前記２項に記載の方法。

（４）　前記マップファイルが、前記第１のソフトウエアプログラムの前記バージョンのインターフェースを構成する大域記号をさらに指定し、前記生成するステップが、前記バージョン設定されたオブジェクトを生成するために、前記マップファイルに従って、前記バージョン名、および、前記バージョンのインターフェースを構成する大域記号を示す情報を前記第１のオブジェクトコードに付加するステップを含む前記１項に記載の方法。

（５）　前記判定するステップが、どの大域記号が前記第２のソフトウエアプログラムに必要とされるのかを調べることによって、前記第２のソフトウエアプログラムに必要とされる前記第１のソフトウエアプログラムのバージョンを判定し、さらに、前記バージョン設定されたオブジェクトにおける情報をチェックすることによって、前記必要とされる大域記号がどのバージョンに存在するのかを判定するステップを含む前記２項に記載の方法。

（６）　前記第１のオブジェクトコードに付加される情報がバージョン定義セクションである前記１項に記載の方法。

（７）　前記第１のオブジェクトコードに付加される情報がバージョン記号セクションである前記１項に記載の方法。

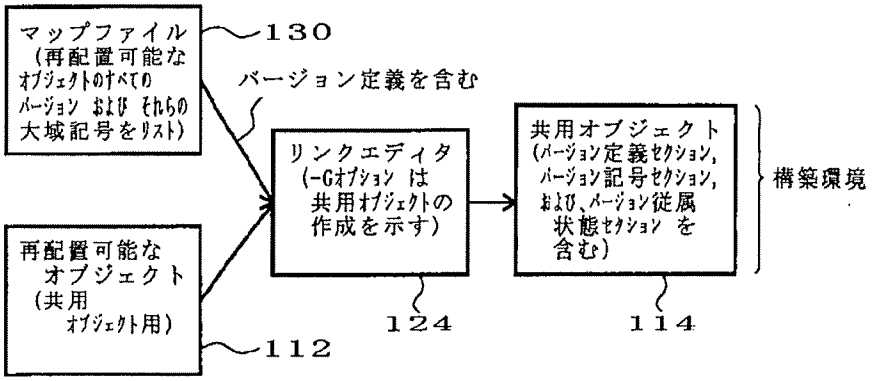
（８）　前記第１のオブジェクトコードに付加される情報がバージョン従属状態セクションである前記２項に記載の方法。

（９）　前記バージョン設定されたオブジェクトが、再

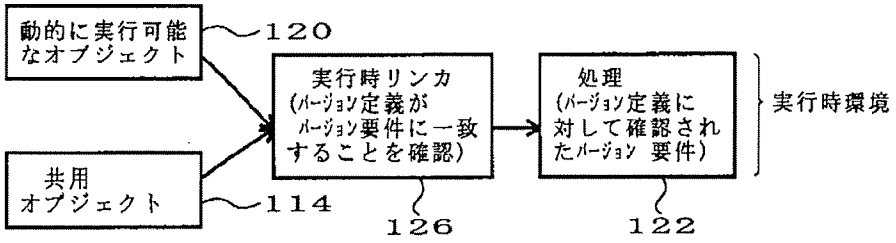
配置可能なオブジェクトである前記１項に記載の方法。
（１０）　前記バージョン設定されたオブジェクトが、動的に実行可能なオブジェクトである前記１項に記載の方法。
（１１）　前記バージョン設定されたオブジェクトが、共用オブジェクトである前記１項に記載の方法。
【００６６】（１２）　ソフトウエアプログラムにバージョン設定情報を付す装置であって、第１のソフトウエアプログラムのための第１のオブジェクトコードを格納する記憶媒体と、前記第１のソフトウエアプログラムのバージョンに関するバージョン名を指定するマップファイルを格納する記憶媒体と、前記マップファイルに従って、前記第１のオブジェクトコードに対して、前記第１のソフトウエアプログラムの前記バージョンのバージョン名を定義する付加情報を付し、これにより、バージョン設定されたオブジェクトを生成するリンカとを具備する装置。
（１３）　第２のソフトウエアプログラムのための第２のオブジェクトコードを格納する記憶媒体と、前記第２のソフトウエアプログラムに必要とされるバージョンを示す付加情報を前記第２のオブジェクトコードに付すことによって、前記第２のオブジェクトコードを前記バージョン設定されたオブジェクトにリンクし、これにより、動的に実行可能なプログラムを生成するリンカとをさらに具備した前記１２項に記載の装置。
（１４）　前記動的に実行可能なオブジェクトにおいて指定された前記必要とされるバージョンが、前記バージョン設定されたオブジェクトに定義されたバージョンに一致していると判定した場合、前記動的に実行可能なプログラムの実行を可能にする実行時リンカをさらに具備した前記１３項に記載の装置。
（１５）　前記バージョン設定されたオブジェクトが、再配置可能なオブジェクトである前記１２項に記載の装置。
（１６）　前記バージョン設定されたオブジェクトが、動的に実行可能なオブジェクトである前記１２項に記載の装置。
（１７）　前記バージョン設定されたオブジェクトが、共用オブジェクトである前記１２項に記載の装置。
【００６７】（１８）　動的に実行可能なオブジェクトに必要とされるオブジェクトのバージョンが該実行可能オブジェクトの実行中に存在することを判定させるための、コンピュータによって読み取り可能なコードを格納したコンピュータによって使用可能な媒体を備えたコンピュータプログラム製品であって、コンピュータに、第１のソフトウエアプログラムのための第１のオブジェクトコードを用意させる第１のコンピュータ読み取り可能プログラムコード装置と、コンピュータに、前記第１のソフトウエアプログラムのバージョンに関連したバージョン名を指定するマップファイルを用意させる第２のコンピュータ読み取り可能プログラムコード装置と、前記第１のソフトウエアプログラムのバージョン名を示す情報が、前記マップファイルに従って、前記第１のオブジェクトコードに付加されるよう、コンピュータに、前記第１のオブジェクトコードをリンクさせ、これにより、バージョン設定されたオブジェクトを生成させる第３のコンピュータ読み取り可能プログラムコード装置とを具備したコンピュータプログラム製品。
（１９）　コンピュータに、第２のソフトウエアプログラムのための第２のオブジェクトコードを用意させる第４のコンピュータ読み取り可能プログラムコード装置と、コンピュータに、前記第２のソフトウエアプログラムに必要とされる前記第１のソフトウエアプログラムのバージョンを判定させ、前記第２のソフトウエアプログラムに必要とされるバージョンを示す情報を前記第２のオブジェクトコードに付加させることによって、前記バージョン設定されたオブジェクトに対する前記第２のオブジェクトコードのリンクを実行させる第５のコンピュータ読み取り可能プログラムコード装置とをさらに具備した前記１８項に記載のコンピュータプログラム製品。
（２０）　コンピュータに、前記動的に実行可能なプログラムがプログラムを実行するために必要とされるバージョンが前記バージョンを有するオブジェクトに定義されたバージョンと一致しているか否かを判定させる第６のコンピュータ読み取り可能プログラムコード装置をさらに具備し、前記動的に実行可能なプログラムが前記バージョン設定されたオブジェクトの前記必要とされるバージョンを呼出すことを特徴とする前記１９項に記載のコンピュータプログラム製品。
【００６８】
【発明の効果】以上のように、この発明は、オブジェクトの新たなバージョンが作成される毎に該オブジェクトの名前を変える必要性を無くし、且つ、バージョン変更に対処するために必要な部分のみのアップグレードを可能にするので、簡単且つ効率的なバージョン設定を実現できる、という優れた効果を奏する。
【図面の簡単な説明】
【図１】この発明に係るコンピュータシステムを示すブロック図。
【図２】構築時における図１のリンクエディタの入出力を示す図。
【図３】構築時における図１のリンクエディタの他の入出力を示す図。
【図４】構築時における図１のリンクエディタの他の入出力を示す図。
【図５】実行時における図１の実行時リンカプログラムの入出力を示す図。
【図６】オブジェクトにバージョン定義セクションおよびバージョン記号セクションを付加するために、図２のリンクエディタによって実行されるステップを示すフロ

ーチャート。
【図７】図６の処理の詳細を示すフローチャート。
【図８】図２のマップファイルのフォーマットを示す図。
【図９】バージョン設定されたオブジェクトに含まれる図２のリンクエディタの出力を示す図。
【図１０】図９のバージョン定義セクションのフォーマットを示す図。
【図１１】図９のバージョン記号セクションのフォーマットを示す図。
【図１２】動的に実行可能なアプリケーションプログラムにバージョン従属状態セクションを付加するために、図３または図４のリンクエディタによって実行されるステップを示すフローチャート。
【図１３】図４のマップファイルのフォーマットを示す図。
【図１４】動的に実行可能なアプリケーションプログラムに含まれる図３のリンクエディタの出力を示す図。
【図１５】図９および図１４のバージョン従属状態セクションのフォーマットを示す図。
【図１６】図１０、図１１および図１５のセクションヘッダのフォーマットを示す図。
【図１７】図１６のヘッダにおける様々な値のリストを示す図。
【図１８】図１６のヘッダにおける様々な値のリストを示す図。
【図１９】アプリケーションプログラムのバージョン要件が該アプリケーションプログラムにリンクされているオブジェクトに存在するバージョンに一致していることを確認するために、図５の実行時リンカによって実行されるステップを示すフローチャート。
【図２０】前記リンクエディタによってオブジェクトに付加されたバージョン定義セクションの一例を示す図。
【図２１】前記リンクエディタによってオブジェクトに付加されたバージョン記号セクションの一例を示す図。
【図２２】前記リンクエディタによってバージョン設定されたオブジェクトおよび動的に実行可能なアプリケーションプログラムの一方または両方に付加可能なバージョン従属状態セクションの一例を示す図。
【符号の説明】
１０２　ＣＰＵ
１０４　メモリ
１１０　ソースコード
１１４　共用オブジェクト
１１６　ソースコード
１１８　再配置可能なオブジェクト
１２０　動的に実行可能なオブジェクト
１２４　リンクエディタ
１２４　実行時リンカ
１３０　マップファイル
１３２　マップファイル

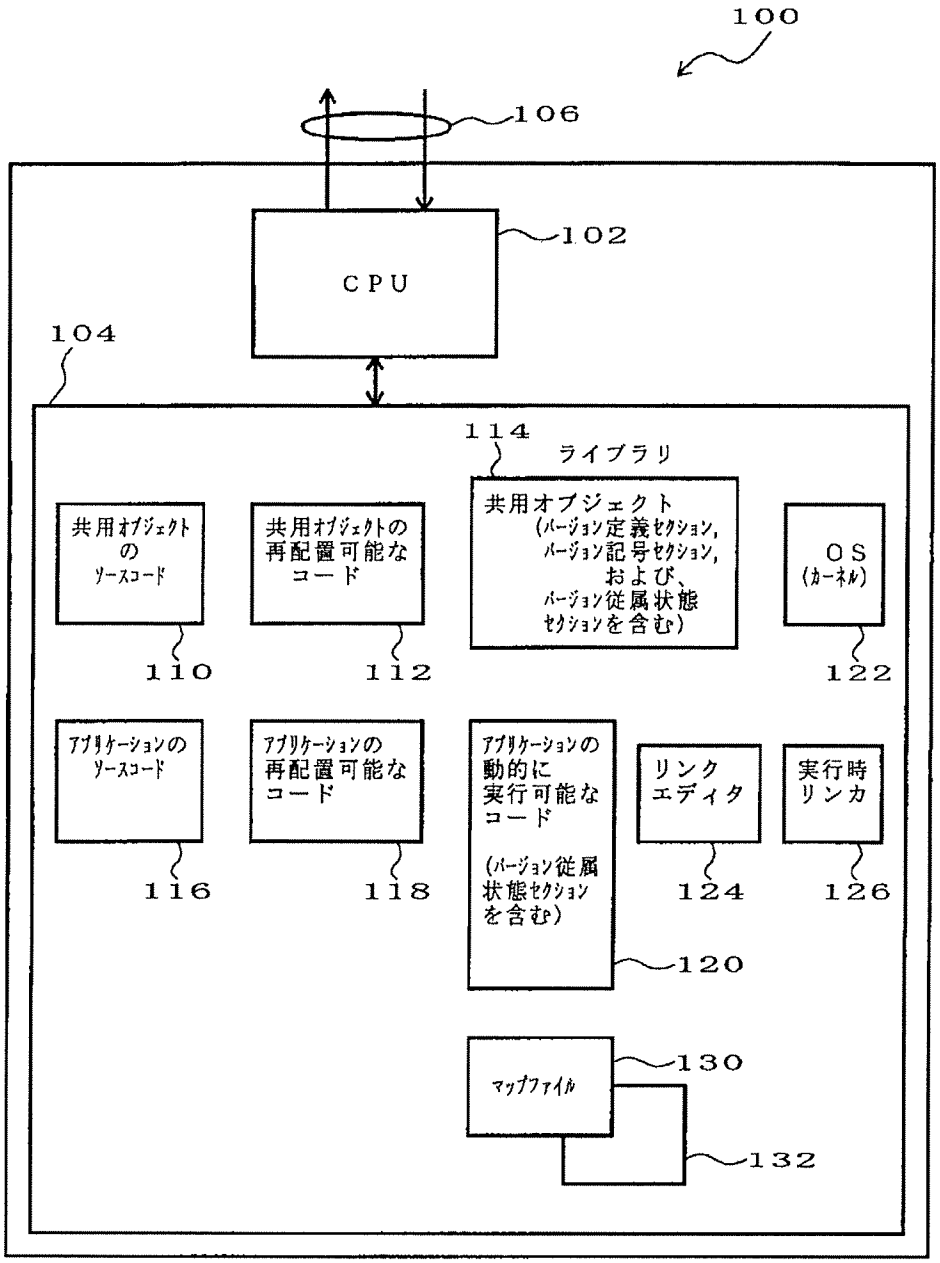
【図2】



【図5】



【図１】



【図８】

```
[<version name>] {
    [<scope>:]
        [<symbol(s)>];
            } [<inheritance information>] ;
```

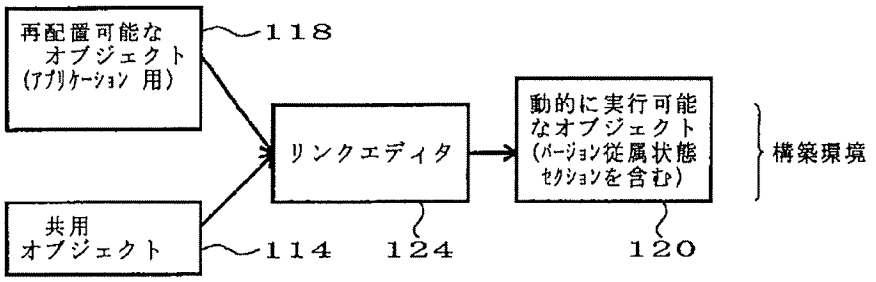
(マップファイルにおける) リンク指示のフォーマット(バージョン定義)

【図１３】

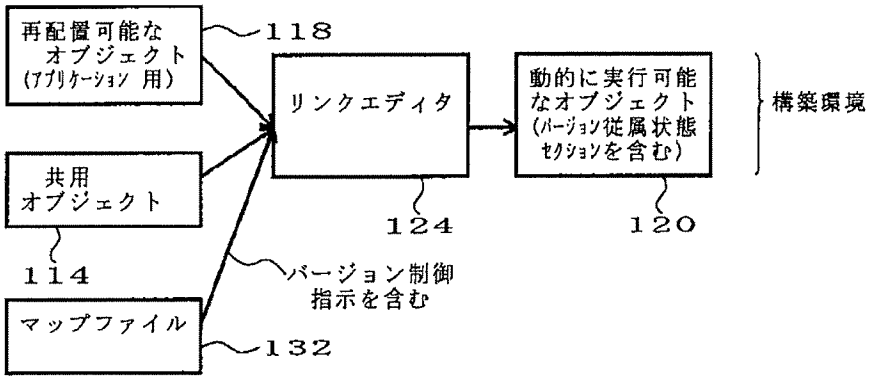
```
<object name>-<version(s)>;
```

リンク指示のフォーマット
(マップファイルにおけるバージョン制御指示)

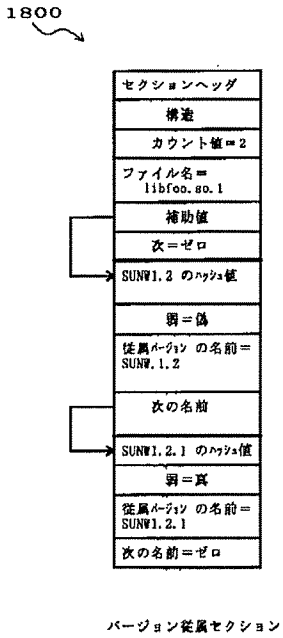
【図3】



【図4】

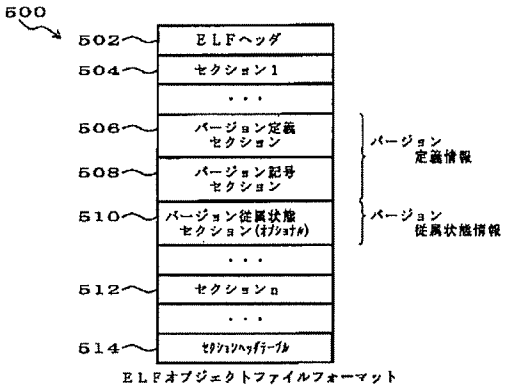


【図22】



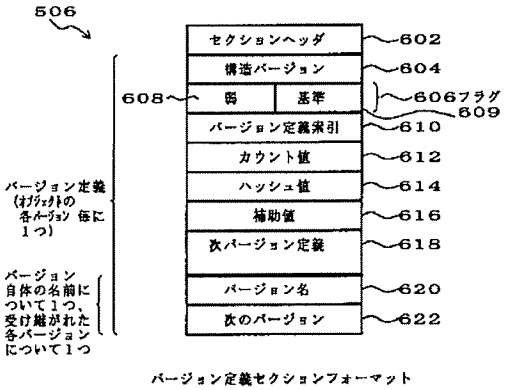
バージョン従属セクション

【図9】



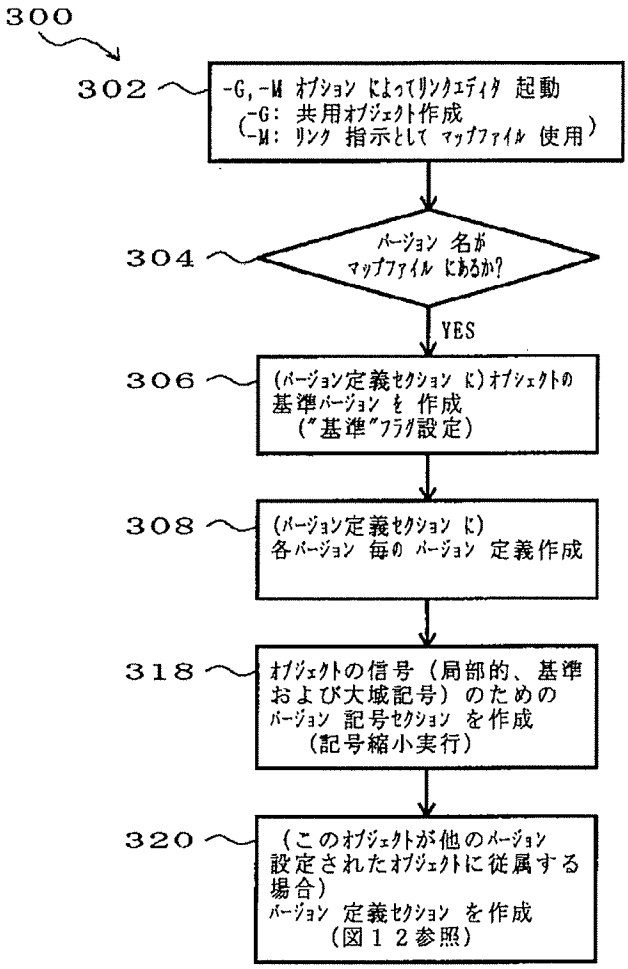
ELFオブジェクトファイルフォーマット

【図10】



バージョン定義セクションフォーマット

【図６】



リンクエディタがバージョン定義セクション
およびバージョン記号セクションを作成する処理

【図１６】

セクションヘッダフォーマット

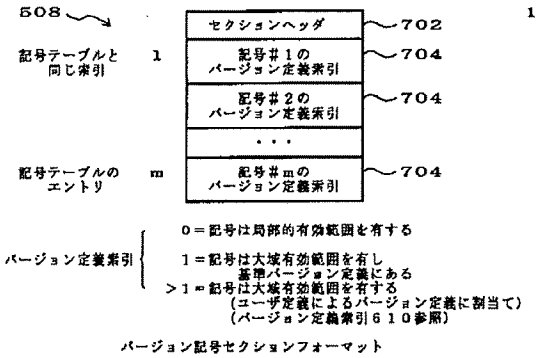
```
typedef struct {
        Elf32_Word      sh_name;    ～1202
        Elf32_Word      sh_type;    ～1204
        Elf32_Word      sh_flags;
        Elf32_Addr      sh_addr;
        Elf32_Off       sh_offset;
        Elf32_Word      sh_size;    ～1306
        Elf32_Word      sh_link;    ～1308
        Elf32_Word      sh_info;    ～1310
        Elf32_Word      sh_addralign;
        Elf32_Word      sh_entsize;
} Elf32_Shdr;
```

【図１７】

セクションタイプ, sh_type

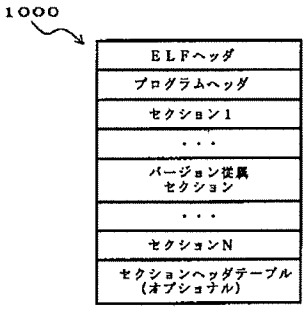
名前	値
SHT_NULL	0
SHT_PROGBITS	1
SHT_SYMTAB	2
SHT_STRTAB	3
SHT_RELA	4
SHT_HASH	5
SHT_DYNAMIC	6
SHT_NOTE	7
SHT_NOBITS	8
SHT_REL	9
SHT_SHLIB	10
SHT_DYNSYM	11
SHT_SUNW_verdef	0x6ffffffd
SHT_SUNW_verneed	0x6ffffffe
SHT_SUNW_versym	0x6fffffff
SHT_LOPROC	0x70000000
SHT_HIPROC	0x7fffffff
SHT_LOUSER	0x80000000
SHT_HIUSER	0xffffffff

【図１１】



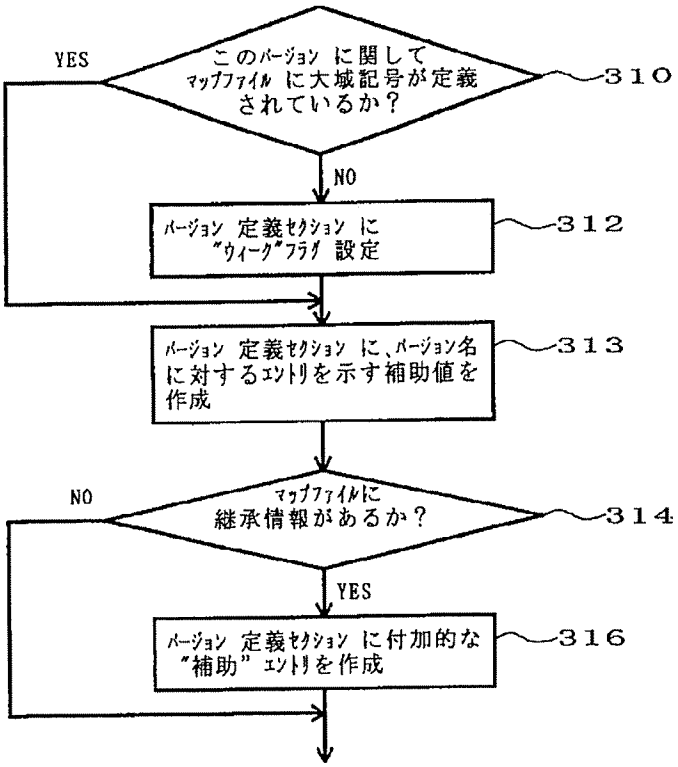
バージョン記号セクションフォーマット

【図１４】



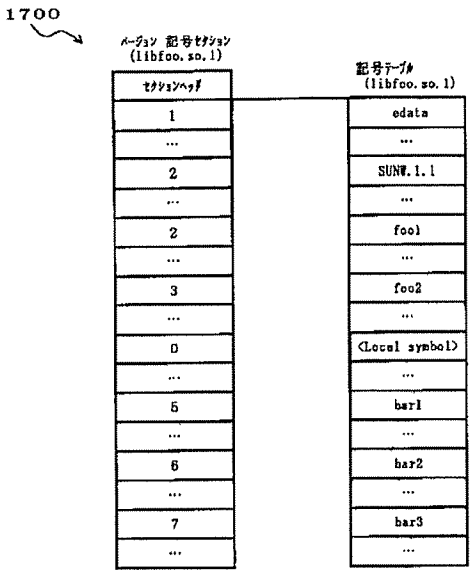
ＥＬＦオブジェクトファイル
(リンクエディタからの動的に実行可能な出力)

【図７】

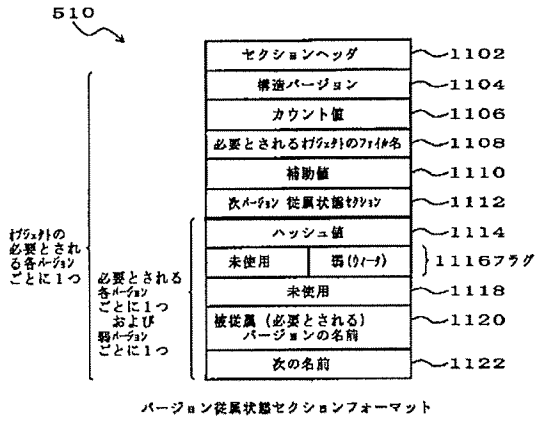


バージョン定義セクション作成の詳細

【図２１】



【図１５】



バージョン従属状態セクションフォーマット

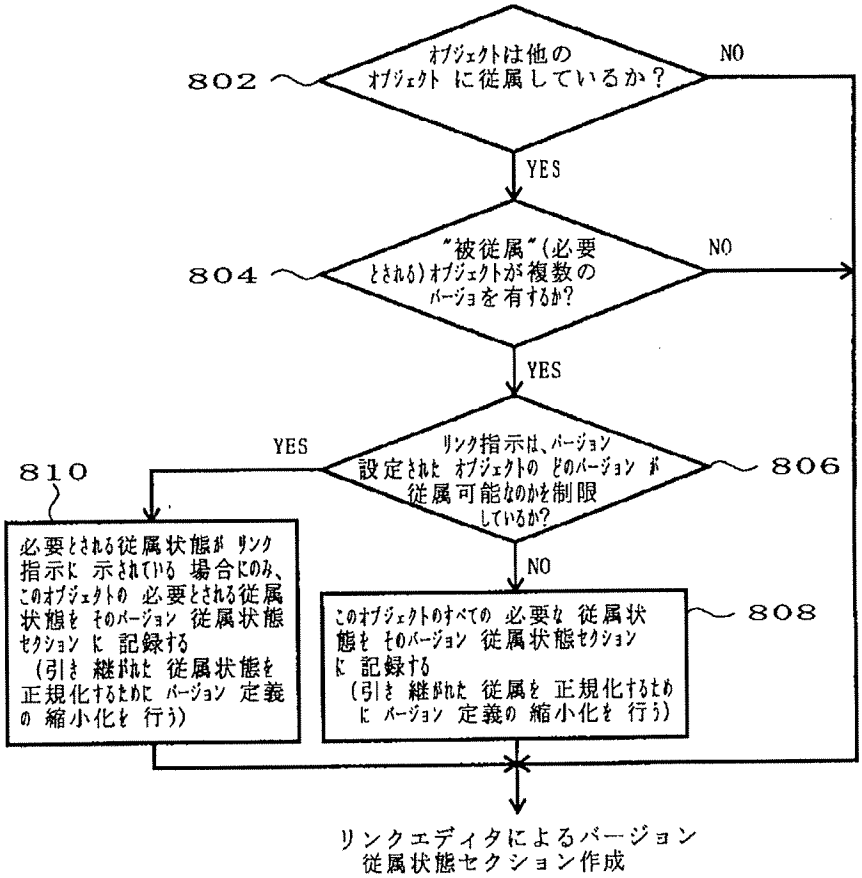
【図１８】

s h_l i n k　および　s h_i n f o　の説明

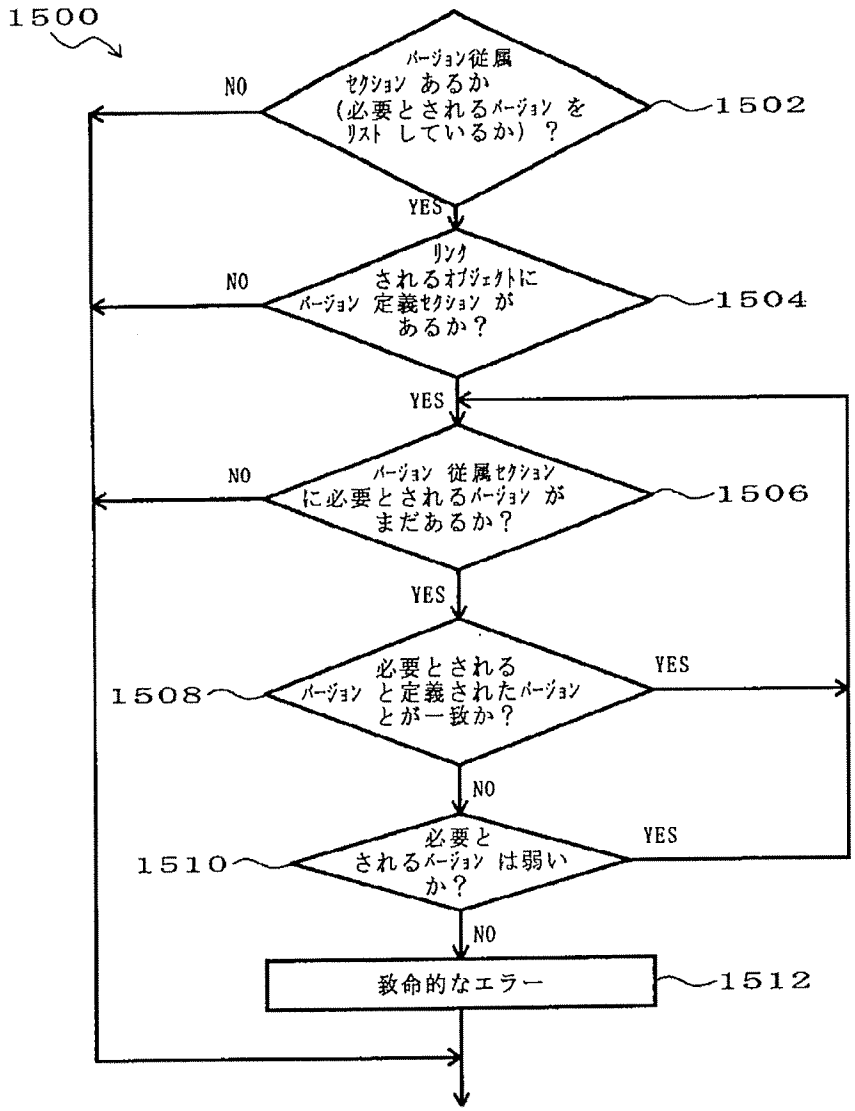
sh_type	sh_link	sh_info
SHT_DYNAMIC	関連ストリングテーブル の セクションヘッダ索引	0
SHT_HASH	関連記号テーブルの セクションヘッダ索引	0
SHT_REL SHT_RELA	関連記号テーブルの セクションヘッダ索引	再配置が適用される セクション の セクションヘッダ 索引
SHT_SYMTAB SHT_DYNSYM	関連ストリングテーブル の セクションヘッダ索引	最後の局部的記号の記号テーブル索引より大きいもの（STB LOCAL 結合）
SHT_SUNW_verdef	関連ストリングテーブル の セクションヘッダ索引	セクション 内のバージョン 定義の数
SHT_SUNW_verneed	関連ストリングテーブル の セクションヘッダ索引	セクション 内のバージョン 従属の数
SHT_SUNW_versym	関連記号テーブルの セクションヘッダ索引	0
その他	SHN_UNDEF	0

1402

【図１２】

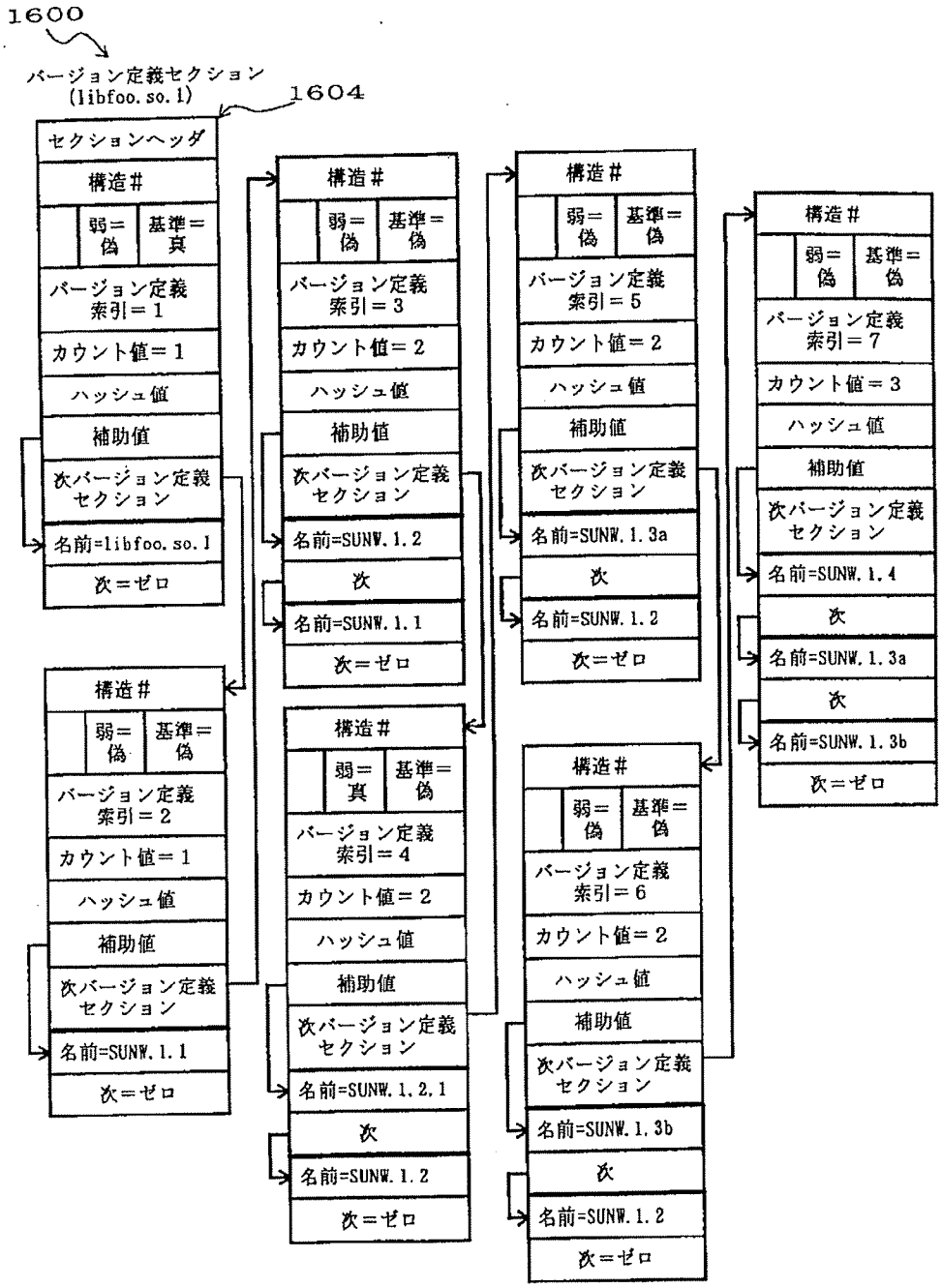

802
オブジェクトは他の
オブジェクト に従属しているか？
NO
YES
804
"被従属"(必要
とされる)オブジェクトが複数の
バージョを有するか?
NO
YES
806
リンク指示は、バージョン
設定された オブジェクトの どのバージョン が
従属可能なのかを制限
しているか?
YES
810
必要とされる従属状態が リンク
指示に 示されている 場合にのみ、
このオブジェクトの 必要とされる従属
状態を そのバージョン 従属状態
セクション に 記録する
(引き 継がれた 従属状態を
正規化するために バージョン 定義
の 縮小化を 行う)
NO
808
このオブジェクトのすべての 必要な 従属状
態を そのバージョン 従属状態セクション
に 記録する
(引き 継がれた 従属を 正規化するため
に バージョン 定義の 縮小化を 行う)
リンクエディタによるバージョン
従属状態セクション作成

【図１９】


1500
バージョン従属
セクション あるか
(必要とされるバージョン を
リスト しているか) ?
1502
NO
YES
リンク
されるオブジェクトに
バージョン 定義セクション が
あるか?
1504
NO
YES
バージョン 従属セクション
に必要とされるバージョン が
まだあるか?
1506
NO
YES
必要とされる
バージョン と定義されたバージョン
とが一致か?
1508
YES
NO
必要と
されるバージョン は弱い
か?
1510
YES
NO
致命的なエラー
1512

実行時確認

【図２０】



フロントページの続き

(72)発明者　ロバート　エー　ジンゲル
アメリカ合衆国　94087　カリフォルニア，
サニーベイル，ライトアベニュー，1377